```cpp
#include <iostream>

using namespace std;
// I have assigned these opcodes to the instructions
const int add=1;
const int sub=2;
const int load=3;
const int store=4;
const int halt=5;

class Computer
{
    private:
        int mem [100];
        int acc;
        int pc;
    public:
        Computer()
        {
            acc=0;
            pc=0;
            for(int i=0;i<100;i++)
                mem[i]=0;

            mem[40] = 57;
            mem[41]=2;
            mem[39]=1;
            mem[42]=60;

        }

        void loadprogram(const int * program ,int proglen)
        {
            for(int i=0;i<proglen;i++)
            {
                mem[i*3]=program[i*3];  // this is for opcode
                mem[(i*3)+1]=program[(i*3)+1]; // this is for operand number 1
                mem[(i*3)+2]=program[(i*3)+2]; // and this is for operand 2
            }
        }

        void executeIns(const int * ins)
        {
            if(ins[0]==1) //add
            {
```

```
                cout << "Adding value from memory location " << ins[1] << " to
accumulator."<<endl;

                acc+=mem[ins[1]];
            }
            else if(ins[0]==2) //subtract
            {
                cout << "Subtracting value from memory location " << ins[1] << "
to accumulator."<<endl;

                acc-=mem[ins[1]];
            }
            else if(ins[0]==3) //load
            {
                if(ins[2]== 1 || ins[2]==0) // load single byte
                {
                    acc= mem[ins[1]];
                }
                else if(ins[2]==2) //load 2 bytes
                {
                    //acc = mem[ins[1]] + (mem[ins[1] + 1] << 8);
                    acc = mem[ins[1]] + (mem[ins[1] + 1]);
                }
            }
            else if(ins[0]==4) //store
            {
                mem[ins[1]]=acc;
            }
            else if(ins[0]==5) //halt
            {
                cout<<"Halting all processes & exiting the program"<<endl;
                exit(0);
            }
             //pc += 3;
        }

        void exenextins()
        {
            int current_ins[3];

            current_ins[0]=mem[pc];
            current_ins[1]=mem[pc+1];
            current_ins[2]=mem[pc+2];

            executeIns(current_ins);
```

```
            pc+=3;
        }

        void print()
        {
            cout<<"Memory contents are : "<<endl;
            for(int i=0;i<100;i++)
            {
                cout<<mem[i]<<" ";
            }
            cout<<"Accumulator : "<<acc<<endl;
        }

        int getpc()
        {
            return pc;
        }

        int getProgramLength()
        {
            int len = 0;
            while (mem[len * 3] != halt)
            {

            len++;
            }
            return len;
        }



};


int main()
{
    int program[] =
    {
        load, 40, 2,    // load value from address 40 to acc
        add, 40, 0,     // add acc + mem 40 ki value
        store, 90, 0, // store that answer at address 90
        load, 42, 0,    // same for below instructions
        sub, 90, 0,
        store, 91, 0,
        halt, 0, 0     // Halt program
```

```
    };
    int programLength = sizeof(program) / sizeof(program[0]) / 3;


    Computer computer;
    computer.loadprogram(program, programLength);

    // Fethc decode cyclde

    while (true)
    {
        cout << "Before execution:\n";
        computer.print();

        computer.exenextins();

        cout << "After execution:\n";
        computer.print();
        if (computer.getpc() >= programLength * 3)
        {
            break; // Exit the loop
        }
    }


    return 0;
}
```

So basically the int program array in the main function contains the instructions that are to be executed in the order we need to. So load basically loads the number at that address in our case 40 to the accumulator and then the add 40 adds the accumulator + the number at address 40.

Load has 2 types one is load address,0/1 and the other is load address,2, the 2 means that read a word instead a byte from memory. Store instruction stores the value of the accumulator at the specified address. Halt command basically just shuts down the computer.

So whatever commands u need to execute we need to write them down in the program array.